Neurocomputing 122 (2013) 470–479

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

A cellular learning automata model of investment behavior in the stock market

Milad Mozafari*, Rahim Alizadeh

Department of Mathematics, Shahed University, P.O. Box 18151-159, Tehran, Iran

ARTICLE INFO

Article history:
Received 14 March 2013
Received in revised form
11 June 2013
Accepted 14 June 2013
Communicated by W.S. Hong
Available online 30 June 2013

Keywords:
Cellular automata
Learning automata
Stock market
Investment behavior

ABSTRACT

In this paper, we present a cellular learning automata model for the investment behavior in the stock market. In this model, investors decide to hold, buy, or sell the stocks based on the evolution rules and learn how much they can trust on other investors based on the learning rules. We analyze the effects of imitation, reliability and macrofactors on the stock market and compare the obtained results with the previous approach that is based on cellular automata.

© 2013 Elsevier B.V. All rights reserved.

1. Introduction

Stock markets are complex systems, which are composed of a large number of nonlinear-coupled subsystems. Many scholars have been trying to discover their mechanism and explain the principle of these markets behaviors. In these systems, investors use their own analysis and formulas for investing and decide according to their own perception of the events. In all of this analysis, they consider behaviors of other investors and events about the stock. Besides physical parameters, psychological factors affect the macroeconomic behavior and cause crisis in finance markets. These various factors make impossible the accurate prediction in a stock market. Until now, different approaches for modeling different parts of stock markets have been studied. One of these approaches is Genetic Programming (GP). Genetic Programming is a generalization of genetic algorithms and tries to evolve a population of programs to the new generation of programs in order to get better results. Genetic operators such as crossover and mutation are also available here for evolving a population and a fitness function (measure) is needed to select individuals for the next generation. The general idea behind genetic programming is to find a combination of functions that will give the best results. For example, the authors in [14] used two different methods of GP: Multi-Expression Programming (MEP) and Linear Genetic Programming (LGP) for the prediction of stock index (index of market prices of a particular group of stocks). MEP uses a linear representation of chromosomes and it has the ability of storing multiple solutions for a problem in one chromosome, and LGP acts on linear genomes that consist of expressions of an imperative language. They also used the ensemble of these methods to improve the accuracy of results in their paper; every chromosome represents one trading rule and each gene stands for a parameter value of a trading rule. Another useful tool for modeling complex systems is Artificial Neural Network (ANN). ANNs are inspired from human neural system and try to simulate brain abilities. In these networks, nodes are considered as neurons and arcs between them, as input or output channels for neuron signals; these channels are weighted so that the impact of signals on each neuron can be controlled. ANNs can handle lots of parameters with complex dependencies and their accuracy can be improved by learning (updating and tuning weights) from training samples. There are several types of ANNs such as feed forward networks, radial basis function (RBF) networks, multilayer perception (MLP), etc. that can be used in different types of problems. In [9], performance and accuracy of multi-layer feed forward, generalized, and probabilistic networks in predicting stock returns have been compared against classical linear regression methods and the predictive relationships between different economic variables have been evaluated by information gain techniques in machine learning for data mining. You can find other works using ANNs in [8,22,32,33]. Decision trees are other kinds of tools that can model dynamic and complex decision processes. Decision trees are trees in which nodes are decision

* Corresponding author. Tel.: +98 919 2182516.
E-mail addresses: milad.mozafari69@outlook.com, milad.mozafari69@gmail.com (M. Mozafari), alizadeh@shahed.ac.ir (R. Alizadeh).

0925-2312/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.neucom.2013.06.002

points (subjects), branches are decision processes and leaves are outcomes or results of decision processes. They can be used as a decision support system in lots of decision required problems. For example, the authors in [28] used a two-layer bias decision tree in order to first, increase the accuracy of purchasing stocks and second, improve the investment returns. There are some other tools to be mentioned here such as Neuro-Fuzzy Systems [24], Social Network Systems [4], Quantum Mechanics [37], and Cellular Automata (CA) [12,30].

One of the main tools for investigating the complexity in a stock market is Swarm, an Artificial Stock Market, based on the agent and simulation tool, which is developed by Santa Fe Institute [23]. In the Swarm system, the basic unit of simulation is the swarm, "a collection of agents executing a schedule of events". Swarm accommodates multi-level modeling approaches in which agents can be composed of swarms of other agents in nested hierarchies. It provides general-purpose utilities for designing, implementing, running, and analyzing such multi-agent systems [23]. If the simulation space in the model is a discrete grid, and any agent occupies a fixed cell, the model becomes a cellular automata.

Cellular automata are complete discrete and local grid dynamic systems, which are used to investigate self-organization in statistical mechanics, and especially suitable for complex systems. It has many applications in computer games and modeling physical [27], biological [10], and social processes. With the application of new algorithms to cellular automata, some new types of cellular automata such as Genetic Cellular Automata [16,38], Fuzzy Cellular Automata [7], Neural Nets Cellular Automata [19], Hierarchical Cellular Automata [3], Pushdown Cellular Automata [18] and Cellular Learning Automata (CLA) have been appeared. The interest in studying the economy with cellular automata is rising and some review articles have appeared [4,9,12,15,17,22,24,28–30,38].

As in [12,30], in this paper, we use a three-state (buy/sell/hold) model of the stock market. The difference is that the introduced cellular automata in [12,30] are replaced by new cellular learning automata that have two more following important properties which place our model close to a more real investment process.

According to the previous decisions of the investors during the investment process, some learning rules are added to the model. These penalty–reward rules endow the model with a memory, which helps the investors to make decisions that are more accurate than the previous model. In other words, the new added learning rules increase the accuracy of the evolution rules and complete them. In our model investors trust their neighbors based on their previous decisions and a neighbor that has had more correct predictions is considered more reliable.

In some of our simulations, the constant value of the macrofactors parameter has been replaced by a normal random walk. This highlights the effects of the macrofactors variations, during the investment process.

The structure of the remainder of this paper is as follows:

First, in Section 2, we describe some preliminary concepts, which will be used in the next sections. In Section 3, based on the model in [30], a new cellular learning automata model for the investment behavior in the stock market is presented and the details of the model are described. Simulation results are presented in Section 4 and conclusions are highlighted in Section 5.

2. Preliminaries

2.1. Cellular automata

These kinds of automata are arrays of cells with a specified dimension and there is a finite set of states for the cells. In our model, we use a 2-dimensional cellular automata. Cellular automata have a step-by-step behavior and the state of a cell in a step is a function (rules of evolving) of its neighbor cells' state in that step. For more details, we refer to [32].

2.2. Learning automata

Learning automata are machines that can perform a set of actions with different probabilities. The term learning is because they can learn how to perform and react in future steps. With each action they perform, they catch a feedback as a penalty or reward, and this feedback will affect on their subsequent performances (action selection probabilities). Fig. 1 shows the interaction between a learning automaton and its environment, where $\alpha(n)$ denotes the action selected by the automaton.

For more applications and details, see [20,21,25,34–36] and the references therein.

2.3. Cellular learning automata

A combination of cellular automata and learning automata is a cellular learning automata. In this kind of automata, each cell acts as a learning automaton and its reward or penalty depends on defined rules on neighbor cells. This model has learning capability of learning automata and collective behavior and locality of cellular automata [1]. A d-dimensional CLA is a quintuple $CLA = (Z^d, \phi, A, N, F)$ in which

- Z^d is a d-dimensional grid of cells.
- ϕ is a finite set of states that each cell can possess.
- A is a set of learning automata, that each of them assigned to a specific cell.
- $N = \{X_1, \ldots, X_m\}$ is a finite subset of Z^d that is called neighborhood vector.
- $F: \phi^m \rightarrow \beta$ is the local rule of the CLA. β is a set of valid reinforcement signals that can be applied to learning automata.

Cellular learning automata divide into two main groups: synchronous and asynchronous [6]. In a synchronous CLA, different cells are activated synchronously. We use this type of CLAs to construct our model.

CLA tool can model and simulate complex and discrete systems and has many applications in solving different problems such as Numerical Optimization [26], Job Scheduling [1], Skin Detection [2] and Deployment Strategy for Mobile Wireless Networks [11].

2.4. Random walk

In the probability theory, a random walk is a stochastic process in which the change in the random variable is uncorrelated with past changes. Hence, the change in the random variable cannot be forecasted. A Gaussian random walk is a random walk having a step size that varies according to a normal distribution. The increment that a Gaussian random walk makes over a time interval of length h is distributed normally with mean 0 and variance h. More precisely,

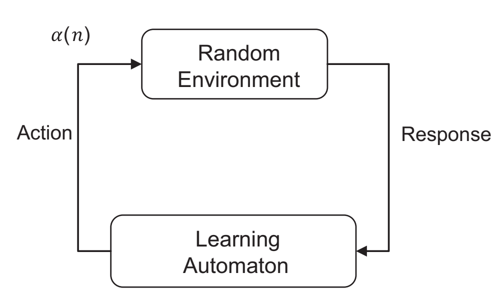

Fig. 1. Interaction between environment and learning automaton [1].

Download English Version:

https://daneshyari.com/en/article/410145

Download Persian Version:

https://daneshyari.com/article/410145

Daneshyari.com